НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

«До захисту допущено»
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
“___”_____201_ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.040301 «Прикладна математика» _____
(код і назва)

на тему: Приховані канали передачі даних в ДСТУ 4145-2002_____
_____
_____

Виконав (-ла): студент (-ка) __4__ курсу, групи ___ФІ-52_____
(шифр групи)

Архипська Єлизавета Андріївна _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник к. т. н. Кучинська Н.В._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2019року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Фізико-технічний інститут

Кафедра математичних методів захисту інформації

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки - 6.040301 «Прикладна математика»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою

М.М.Савчук

_____
(підпис) (ініціали, прізвище)

« 13 »_____02_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

_____ Архипській Єлизаветі Андріївні _____
(прізвище, ім'я, по батькові)

1. Тема роботи Приховані канали передачі даних в ДСТУ 4145-2002 _____
_____ ,
керівник роботи к. т. н. Кучинська Н.В. _____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 27.05.2019 р. № 1414-С

2. Термін подання студентом роботи 14.06.2019

3. Вихідні дані до роботи алгоритми формування прихованих каналів в ДСТУ 4145-2002 і приклади їх реалізації. _____
_____
_____
_____

4. Зміст роботи У цій роботі представлено алгоритми побудови прихованих каналів передачі даних в сучасному стандарті цифрового підпису ДСТУ 4145-2002 _____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) презентація _____

_____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання вересень 2018

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд опублікованих джерел за	вересень 2018-	
	заданою темою	листопад 2018	
2.	Дослідження алгоритму формування	листопад 2018-	
	цифрового підпису в ДСТУ 4145-2002	грудень 2018	
3.	Побудова широкосмугового каналу в	грудень 2018-	
	ДСТУ 4145-2002	січень 2019	
4.	Вузькосмугову канали в ДСТУ 4145-	лютий 2019	
	2002	-березень 2019	
5.	Реалізація побудованих прихованих каналів	квітень 2019	

Студент _____ Архипська Є.А
(підпис) (ініціали, прізвище)

Керівник роботи _____ Кучинська Н.В.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Кваліфікаційна робота містить: 79 стор., 13 джерел і 1 додаток.

У даній роботі було досліджено можливості побудови та використання прихованих каналів в алгоритмі цифрового підпису ДСТУ 4145-2002.

Більшість організацій, які працюють через мережу, мають необхідність підтверджувати свої електронні документи, як звичайний підпис підтверджує паперовий документ. Для цього використовується електронний цифровий підпис, який гарантує цілісність і підтверджує авторство документу. ДСТУ 4145-2002 – нині діючий національний стандарт України, що описує механізм формування та перевірки електронного цифрового підпису основаного на властивостях груп точок еліптичних кривих над полями $GF(2^m)$ та правилах застосування цих механізмів до повідомлень, що пересилаються каналами зв'язку та/або обробляються у комп'ютеризованих системах загального призначення. Застосування цього стандарту гарантує цілісність підписаного повідомлення, автентичність його автора та неспростовність авторства.

У даній роботі було отримано алгоритми побудови широкосмугового і вузькосмугових каналів, досліджено їх переваги і недоліки.

Темою роботи є приховані канали передачі інформації в ДСТУ 4145-2002.

Метою дослідження є знаходження прихованих каналів передачі інформації в ДСТУ 4145-2002.

Об'єктом дослідження є процес прихованої передачі інформації при використанні алгоритмів цифрових підписів.

Предметом дослідження є формування прихованих каналів в алгоритмі цифрового підпису ДСТУ 4145-2002.

Актуальність даного дослідження полягає у тому, що досліджується нині діючий стандарт цифровго підпису на еліптичних кривих ДСТУ 4145-2002 на предмет наявності прихованих каналів, які можуть порушувати

політику безпеки.

Практичне значення результатів полягає полягає у тому, що вони можуть бути використані для побудови ефективних алгоритмів ліквідації прихованих каналів в ДСТУ 4145-2002, а також для розробки протоколів узгодження сеансових ключів на прихованих каналах.

ЦИФРОВИЙ ПІДПИС, ДСТУ 4145-2002, ПРИХОВАНИЙ КАНАЛ, ЕЛІПТИЧНА КРИВА

ABSTRACT

The thesis contains: 79 pages, 13 sources and 1 appendix.

In this thesis, the possibilities of constructing subliminal channels in DSTU 4145-2002 were investigated.

Most organizations that work through the network need to confirm their electronic documents, as a normal signature confirms a paper document. For this, an electronic digital signature is used to guarantee the integrity and confirmation of the authorship of the documentation. DSTU 4145-2002 – the current national standard of Ukraine describing the mechanism for the formation and verification of an electronic digital signature based on the properties of groups of points of elliptic curves over fields $GF(2^m)$ and the rules for applying these mechanisms to messages sent by channels and / or processed in computerized general-purpose systems. The application of this standard guarantees the integrity of the signed message, the authenticity of its author and the authorship of the authorship.

In this paper algorithms for building broadband and narrowband channels were obtained, their advantages and disadvantages were studied.

The theme of the thesis is the subliminal channels of information transmission in DSTU 4145-2002.

The object of research is the process of subliminal information transmission using the algorithms of digital signatures

The purpose of the thesis is to find the subliminal channels of information transmission in DSTU 4145-2002.

The relevance of this study is that the current standard of digital signature on the elliptic curves of DSTU 4145-2002 is being investigated for the presence of hidden channels that may violate the security policy.

Practical value of the results lies in the fact that they can be used to construct effective algorithms for eliminating subliminal channels in DSTU 4145-2002, as well as for developing protocols for the coordination of session

keys on subliminal channels.

DIGITAL SIGNATURE, DSTU 4145-2002, SUBLIMINAL CHANNEL, ELLIPTIC CURVE

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЦП —цифрововий підпис

ЕЦП — електронний цифровий підпис

ПВП — псевдовипадкова послідовність

ГПВП — генератор псевдовипадкових послідовностей

ЕК — еліптична крива

ВСТУП

Актуальність дослідження. Актуальність даного дослідження полягає у тому, що досліджується нині діючий стандарт ЦП на еліптичних кривих ДСТУ 4145-2002 на предмет наявності прихованих каналів, які можуть порушувати політику безпеки.

Метою дослідження є знаходження прихованих каналів в ДСТУ 4145-2002. **Задача дослідження** полягає у побудові прихованих каналів. Для розв'язання задачі необхідно вирішити такі завдання:

1) дослідити алгоритм формування і перевірки ЦП в ДСТУ 4145-2002;

2) побудувати алгоритм формування широкосмугового каналу в ДСТУ 4145-2002;

3) побудувати алгоритми формування детермінованого і ймовірнісного каналів в ДСТУ 4145-2002;

Об'єктом дослідження є процес прихованої передачі інформації при використанні алгоритмів ЦП.

Предметом дослідження є формування прихованих каналів в алгоритмі ЦП ДСТУ 4145-2002.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: абстракна алгебра, теорія чисел, теорія ймовірностей.

Наукова новизна отриманих результатів полягає у тому, що вперше запропоновано алгоритми побудови прихованих каналів в ДСТУ 4145-2002, розглянуто їх переваги і недоліки, приведено приклади побудови.

Практичне значення результатів полягає полягає у тому, що вони можуть бути використані для побудови ефективних алгоритмів ліквідації прихованих каналів в ДСТУ 4145-2002, а також для розробки протоколів узгодження сеансових ключів на прихованих каналах.

Апробація результатів та публікації. Результати цієї роботи були частково представлені на XVII Науково-практичній конференції студентів,

аспірантів та молодих вчених "Теоретичні і прикладні проблеми фізики, математики та інформатики"(26-27 квітня 2019р., м. Київ).

1 ПРИХОВАНІ КАНАЛИ В АЛГОРИТМАХ ЦП

У даному розділі йде мова про загальні відомості про приховані канали, їх класифікацію, виявлення у відомих алгоритмах ЦП.

1.1 Загальні відомості про приховані канали

Прихованний канал — це непередбачений розробником комунікаційний канал, по якому може передаватися інформація, порушуючи політику безпеки інформації. Вперше поняття прихованого каналу було введено американським вченим в галузі теорії обчислюваних систем Батлером Лемпсоном у роботі [1], де він запропонував наступне визначення: «приховані канали — це ті канали, які зовсім не розроблені для передачі інформації, такі як вплив службової програми на завантаження системи».

Сутність прихованного каналу полягає у тому, що він прихований від засобів розмежування доступу, систем виявлення вторгнень навіть безпечних операційних систем, які контролюють тільки інформаційні потоки. Прихований канал не може бути виявлений або проконтрольований апаратними механізмами забезпечення безпеки, тому що він не використовує встановлені системою механізми передачі і обробки даних, такі як читання і запис. В реальних системах достатньо важко розпізнати приховані канали, можливо лише зменшити ступінь загрози безпеки інформації.

За допомогою прихованих каналів можуть бути реалізовані наступні порушення політики безпеки:

1) загроза впровадження шкідливих програм і даних;

2) загроза передачі певних команд для агентів;

3) загроза витоку криптографічних ключів, паролів та інших інформаційних об'єктів.

Приховані канали можуть використовуватись як для забезпечення анонімності і конфіденційності, так і для організації витоку даних. На практиці виявлення прихованої передачі інформації є однією з найскладніших задач інформаційної безпеки. У роботі [2] Сіммонс представив концепцію прихованого каналу в звичайних схемах цифрового підпису. Приховане повідомення виглядає як звичайний цифровий підпис і лише авторизований одержувач може прочитати його. Прихований канал у цифровому підписі має кілька додатків [3]. Наприклад, постачальник кредитної картки може приховати кредит власника картки, історію та кредитний ліміт у цифровому підписі для кредитної картки.

Окрім роботи Сіммонса, в 1997 році Харн і Гонг запропонували дві схеми цифрового підпису з реалізацією широкосмугового (див. пункт 1.2) прихованого каналу, який не вимагав присутності отримувача для розділення секретного ключа. Але довжина цифрового підпису генерувалася у запропонованих схемах занадто довго, при чому розміри секретних ключів відправника і отримувача теж були довгими. Пізніше приховані канали були знайдені у всіх важливих схемах підпису, як RSA [4], DSA [5] або ECDSA [6]. Тому забезпечення уникнення підписувача користування підсвідомим каналом виявилося складним завданням.

1.2 Класифікація прихованих каналів

У 1985 році Сіммонс [7] показав, що в будь-якій схемі цифрового підпису, в якій використовується α біт для передачі підпису і β біт для забезпечення безпеки($\alpha > \beta$), решта $\alpha - \beta$ біти потенційно доступні для

прихованого зв'язку.

В [8] Сіммонс дав таку класифікацію прихованих каналів:

• широкосмугові (англ. broadband) – кількість біт в прихованому повідомленні порівняна з числом біт у випадковому числі. Зазвичай приховане повідомлення і є цим випадковим числом;

• вузькосмугові (англ. narrowband) — кількість біт в прихованому повідомлення набагато менше кількості біт у випадковому числі. У такому випадку, як правило, довжина повідомленння залежить від обчислювальних можливостей підписуючого і перевіряючого.

Сіммонс визначив, що якщо прихований канал використовує всі або майже всі біти $\alpha - \beta$, він є широкосмуговим, тоді як якщо він використовує лише частину $\alpha - \beta$ біт, він є вузькосмуговим.

Не зважаючи на максимальну пропускну здатність у більшості випадків широкосмугові канали мають суттєвий недолік, в результаті якого перевіряючому стає відомий секретний ключ підпису. Також через певні умови, накладені на випадкове число, не будь-яке повідомлення можна підставити замість цього випадкового числа.

Вузькосмугові канали не потребують розкриття ключа підпису. Але для їх організації потрібно багато обчислювальних ресурсів – пам'яті і процесорного часу. Низька пропускна здатність каналу визначається обмеженістю цих ресурсів.

Візькосмугові канали поділяються на ймовірнісні і детерміновані. У ймовірнісних каналах приховане повідомлення можна отримати з деякою ймовірністю, яка не дорівнює 1, при чому чим більше біт в повідомленні – тим менша ймовірність. В детермінованих каналах приховане повідомлення можна обчислити завжди, але процес обчислення потребує велику обчислювальну потужність.

1.3 Приховані канали в системах цифрового підпису

Розгленемо виявлені приховані канали в алгоритмі ЦП з відкритим ключем DSA та алгоритмі ЦП на еліптичних кривих ГОСТ Р 34.10-2001

1.3.1 DSA

У 1991 році Національним Інститутом Стандартів і Технологій(США) було запропонувано алгоритм цифрового підпису DSA на основі схеми Ель-Гамаля, який спочатку був запатентований, але наразі доступний для використання без ліцензійних відрахунків. Для підпису повідомлення необхідна пара ключів – публічний і секретний, при чому секретний ключ має бути відомий тільки особі, підписуючій повідомлення, а публічний – будь-кому, хто хоче перевірити підпис. Також публічними є параметри самого алгоритму ЦП.

Розглянемо алгоритм Формування і перевірки ЦП, який містить 4 основні кроки [8], [9].

1) Генерація публічних ключів p,q і g:

– велике просте число p (512-1024 біти) обирається випадковим чином і має задовольняти умові: $p-1$ ділиться на 160-бітне просте число q;

– елемент g поля $GF(p)$ знаходиться вибором елемента $z \in GF^*(p)$ для якого $g \equiv z^{(p-1)/q} mod p > 1$.

2) Генерація секретних ключів:

– користувач обирає випадковий елемент $x \in GF^*(p)$, який є ключем аутентифікації;

– користувач публікує $y \equiv g^x \bmod p$ як ключ верифікації .

3) m – повідомлення для підпису, $H(\cdot)$ – функція гешування,

визначена для DSA .Підписаним повідомленням є трійка$(m; r, s)$. Підпис:

– обчислюється геш повідомлення $h = H(m)$;

– обирається випадкове число $k \in GF^*(q)$ (ключ сессії);

– обчислюється $r \equiv (g^x \bmod p) \bmod q$;

– обчислюється $s \equiv k^{-1}(h + xr) \bmod q$, де $kk^{-1} \equiv 1 \bmod q$.

4) Верифікація:

– одержувач обчислює геш повідомлення $h = H(m)$;

– одержувач обчислює $t \equiv s^{-1} \bmod q$;

– одержувач обчислює $u_1 \equiv ht \bmod q$;

– одержувач обчислює $u_2 \equiv rt \bmod q$;

– одержувач обчислює $v \equiv (g^{u_1} y^{u_2} \bmod p) \bmod q$;

– підпис вірний тоді і тільки тоді, коли $r = v$.

Даний алгоритм базується на складності задачі обчислення дискретного логарифмування в скінченному полі.

Широкосмуговий канал

Найпростіший спосіб побудови широкосмугового каналу – поділитися ключем аутентифікації x. Випадкове число $k = m^{'}$ використовується як приховане повідомлення. Без знання x неможливо обчислити $m^{'}$ або зафіксувати використання прихованого каналу.

Відправник замінює k на $m^{'}$ і обчислює $r \equiv (g^{m^{'}} \bmod p) \bmod q$ і $s \equiv m^{'-1}(h + xr) \bmod q$. Зі знанням x перевіряючий може обчислити $m^{'} \equiv s^{-1}(h + xr) \bmod q$.

Важливо зазначити, що якщо $h + xr = 0$, то $s = 0$. В такому випадку отримувач не зможе обчислити $m^{'}$. Сіммонс показав, що ймовірність такої події – $1/q$. Оскільки число q достатньо велике(160 біт), така ймовірність близька до 0.

Вузькосмуговий канал

1) 1-бітний канал, перший варіант реалізації.

Відправнику і одержувачу необхідно створити спільне просте число $P > q$. Публічне число p не слід використовувати, оскільки у такому випадку кожен може перевірити використання прихованого каналу. За

допомогою даного каналу можна приховано передати 1 біт в залежності від того являється r квадратичним лишком або не лишком за модулем P. Одержувач може легко це перевірити за критерієм Ейлера.

Теорема 1.1. *Нехай p-непарне просте число, a – ціле число, взаємнопросте з p. Тоді*

$$a^{\frac{p-1}{2}} = \begin{cases} 1(mod\ p),\ \exists x : a \equiv x^2(modp), \\ -1(mod\ p),\ інакше. \end{cases}$$

Відправник може обирати k доки не отримає $r = (g^k \bmod p) \bmod q$ – квадратичний лишок(нелишок). Ймовірність отримати бажаний результат дорівнює $\frac{1}{2}$, оскільки існує $\frac{(P-1)}{2}$ квадратичних лишків і $\frac{(P-1)}{2}$ квадратичних нелишків.

2) 1-бітний канал, другий варіант реалізації.

Відправник і одержувач створюють спільну випадкову бінарну послідовність $B = b_1, ..., b_t$. Обидва узгоджують позицію в бітовому представленні числа r, яка буде використана для передачі прихованого повідомлення. Далі i-й біт повідомлення в i-му підписі ксориться з i-м випадкоми бітом b_i.

3) l-бітний канал.

Відправник обирає випадковий елемент $k \in GF^*(q)$, обчислює $r^{'} \equiv g^{k^{'}} modp$ і секретно відправляє одержувачу. До початку алгоритму ЦП відправник обчислює $k^* \equiv m^{'} + k^{'} modq$, де $m^{'}$ – приховане повідомлення довжини l. Замість випадкового числа в алгоритмі використовується $k*$. Відправник передає трійку (m,r,s). Знаючи r' одержувач перебирає усі можливі варіанти доки не знайде m', для якого виконується рівність

$$(g^{m'+k'} modp) modq \equiv r \equiv (r^{'} g^{m'} modp) modq.$$

1.3.2 ГОСТ Р 34.10-2001

Російський стандарт ЦП, що базується на еліптичних кривих. Наразі являється застарілим, оскільки йому на заміну введено новий стандарт ГОСТ р 34.10-2012. Стійкість цього алгоритму базується на складності обчислення дискретного логарифму в групі точок еліптичної кривої. Новий стандарт відрізняється наявністю додаткового варіанту параметрів схем (для довжини секретного ключа в 512 біт) і вимогою використовувати функцію гешування ГОСТ Р 34.11-2012, яка замінила функцію гешування ГОСТ Р 34.11-94.

Опишемо загальний алгоритм формування і перевірки ЦП.

1) Параметри схеми цифрового підпису :

– просте число p – модуль еліптичної кривої, $p > 2^{255}$;

– еліптична крива E, яка задається своїм інваріантом $J(E)$ або коефіцієнтами $a, b \in G_p$;

– ціле число m – порядок групи точок еліптичної кривої, $m \neq p$;

– просте число q – порядок деякої циклічної підгрупи групи точок еліптичної кривої, $q|m$ і $2^{254} < q < 2^{256}$;

– точка P еліптичної кривої E – генератор підгрупи порядка q. $qP = O$ і $kp \neq O$ для усіх $k = 1, 2, ..., q - 1$, де точка O – нейтральний елемент групи;

– $h(M)$ – 256-бітна функція гешування.

2) Ключі ЦП:

– ключ підпису – ціле число d, яке задовольняє рівність $0 < d < q$.

– ключ перевірки – точка еліптичної кривої Q, яка задовольняє рівність $dP = Q$.

– додаткові умови:

а) $p^t \neq 1(modq)$, для усіх цілих чисел $t = 1, 2, ..., B$, де $b \geq 31$;

б) $J(E) \neq 0$ або 1728.

3) Формування ЦП:

а) обчислити геш повідомлення $z = h(M)$;

б) обчислити $e = z \pmod q$, якщо $e = 0$ встановити $e = 1$;

в) згенерувати випадкове число k, таке що $0 < k < q$;

г) обчислити точку еліптичної кривої $C = kP$ і за допомогою неї знайти $r = x_C \pmod q$, де x_C – координата x точки C. Якщо $r = 0$, повернутися до попереднього кроку.

д) знайти:

$$s = rd + ke(modq).(1)$$

Якщо $s = 0$, повернутися до кроку (в);

е) сформувати ЦП $= (r \parallel s)$.

4) Перевірка ЦП:

а) відокремити з ЦП числа r і s. Якщо хоча б одна з нерівностей $0 < r < q$ і $0 < s < q$ неправильна , підпис невірний;

б) обчислити геш-функцію від повідомлення $M : z = h(M)$;

в) обчислити $e = z \pmod q$ і якщо $e = 0$, покласти $e = 1$;

г) обчислити $v = e^{-1} \pmod q$;

д) обчислити $z_1 = sv \pmod q$ і $z_2 = -rv \pmod q$;

е) обчислити точки еліптичної кривої $C = z_1P + z_2Q$ і визначити $R = x_C \pmod q$, де x_C – координата x точки C;

ж) якщо виконується рівність $R = r$ підпис правильний, інакше – неправильний.

Опишемо виявлені приховані канали в даному алгоритмі ЦП з [10].

Широкосмуговий канал

В найпростішому випадку можна підставити число k замість прихованого повідомлення. Якщо перевіряючому відомий ключ підпису d, він може обчислити k за формулою:

$$k = e^{-1}(s - rd)(modq),$$

в іншому випадку потрібно вирішити проблему дискретного

логарифмування в групі точок еліптичної кривої. Недоліки даного каналу:

1) Оскільки в рівності для знаходження k є секретний ключ d, перевіряючому необхідно знати ключ підпису. Також в загальному випадку приховане повідомлення є функцією від числа k, тому k може бути обчислене як деяка обернена функція від прихованого повідомлення. Зі значення числа k можливо легко отримати секретный ключ d з рівності:

$$d = r^{-1}(s - ke)(modq).$$

Таким чином обмін великими повідомленнями може призвести до компрометація секретного ключа d.

2) Стійкість алгоритму сильно залежить від стійкості ГПВП, тому що при повторі числа k можна легко обчислити ключ підпису d з системи:

$$\begin{cases} s_1 = rd + ke_1(modq), \\ s_2 = rd + ke_2(modq). \end{cases}$$

Для захисту від цього недоліку використовується прийом "підсолювання"числа k – заміни частини його біт на випадкові. Але є більш кращий метод, який не зменшує пропускну здатність каналу: замість числа k використати $k_i^{'} = k + h_i \pmod q$, де число h_i – випадкове. В такому випадку перевіряючий має обчислити: $k = k_i^{'} - h_i \pmod q$.

3) Приховане повідомлення, яке підставляється замість числа k окрім умови $0 < k < q$, має задовольняти ще декільком: $r \neq 0 \pmod q$ і $s \neq 0 \pmod q$, оскільки такі підписи відкидаються на кроці (1) алгоритму перевірки підпису.

Отже достатньо простий в реалізації широкосмуговий канал має в собі багато недоліків, тому його краще не використовувати для важливих повідомлень.

Вузькосмуговий канал

Наведемо реалізацію детермінованого і ймовірнісного типів

вузькосмугового каналу.

Ймовірнісний канал

Загальний алгоритм побудови ймовірнісного прихованого каналу.

1) Генерується n великих простих чисел, які є секретним ключем K для ключової функції $h_K()$.

2) Ключова функція $h_K()$ ставить у відповідність частині r підпису послідовність M, яка складається із n біт, де i-й біт дорівнює 1, якщо r – квадратичний лишок за модулем i-того просто числа і 0 в іншому випадку.

3) Для кожного великого простого числа формується ЦП за стандартним алгоритмом.

4) Після перевірки підпису отримувач за допомогою секретного ключа K і алгоритму декодування послідовності M обчислює приховане повідомлення.

Детермінований канал

Наведемо загальний алгоритм побудови детермінованого каналу:

1) Підписувач і перевіряючий домовляються про розмір прихованих повідомлень.

2) Підписувач генерує ключ для ПВП і зберігає його в пам'яті. Потім він генерує число n членів цієї ПВП, де n – число повідомлень, які він може відправити по цьому каналу.

3) Для кожного з цих чисел Підписувач обчислює точку C з кроку (4) алгоритму формування ЦП. Всі n отриманих точок відправляються перевіряючому.

4) Підписувач починає формувати підпис за стандартним алгоритмом, але випадкові числа бере з ПВП, відновленої зі збереженого ключа. Замість точки C використовується точка $C' = C + mP$, де $0 < m < 2^n$.

5) Після перевірки підпису отримувач додає до чергової точки C, отриманої на початку, точку P до тих пір, поки не отримає C'. За числом операцій додавання відновлюється приховане повідомлення m.

Перевіряючий не знає випадкові числа з ПВП і не може їх знайти, так

як встановити залежність між цими числами і множиною переданих чисел майже неможливо. Тому перевіряючий не може обчислити секретний ключ d.

1.4 Загальні поняття і визначення

Нехай K – поле.

Визначення 1.1. Еліптичною кривою E над множиною K називається множина точок $(x, y) \in K \in K$, які задовольняють рівність

$$E : a^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, a_i \in K.$$

В ДСТУ 4145 використовуються еліптичні криві над полем $GF(2^m)$, які задаються афінним рівнянням еліптичної кривої в нормальній формі Вейєрштрасса:

$$y^2 + xy = x^3 + Ax^2 + B,$$

де $A, B \in GF(2^m)$, $B \neq 0$, разом з точкою на нескінченності O.

Визначення 1.2. Порядком ЕК називається число всіх її точок (x, y) разом з точкою на нескінченності O.

Визначення 1.3. Порядком точки P ЕК називається найменше натуральне число $m \neq 0$, для якого виконується $mP = O$.

Порядки кривої E і її точок можуть бути як скінченними так і нескінченними.

Визначення 1.4. Суммою двох точок $P = (x_1, y_1)$ і $Q = (x_2, y_2)$ називається точка $R = P+Q = (x_3, y_3)$, обернена до третьої точки перетину ЕК прямою лінією, яка проходить через точки P і Q.

Якщо $Q = -P$, то $R = O$. В іншому випадку координати точки R

обчислюються за наступними формулами:

$$x_3 = (\frac{y_1 + y_2}{x_1 + x_2})^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + A,$$

$$y_3 = \frac{y_1 + y_2}{x_1 + y_2}(x_1 + x_2) + x_2 + y_1.$$

Якщо $x_1 = 0$, то $2P = O$. Інакше для подвоєної точки $R = 2P$ координати обчислюються за формулами:

$$x_3 = x_1^2 + \frac{B}{x_1^2},$$

$$y_3 = x_1^2 + (x_1 + \frac{y_1}{x_1})x_2 + x_2.$$

Найпростішим скінченним полем є скінченне поле $GF(2)$, яке складається з двох елементів 0 і 1. У цьому полі операції додавання й множення виконуються наступним чином: 0+0=0, 0+1=1+0=1, 1+1=0, 0·0=1·0=0·1=0, 1·1=1. Будь-яке скінченне поле $GF(2^m)$ є m – вимірним векторним простором над полем $GF(2)$. Многочлен $f(t)$ степеня m над полем $GF(2)$ є многочлен вигляду

$$f(t) = t^m + f_{m-1}t^{m-1} + ... + f_0,$$

де коефіцієнти многочлена $f_i \in GF(2), i = 0, ..., m - 1$. Операції над такими многочленами виконуються як операції над звичайними многочленами, тільки операції над коефіцієнтами виконуються в полі $GF(2)$.

Многочлен $f(t)$ ненульового степеня називається незвідним над полем $GF(2)$, якщо він ділиться без залишку над цим полем тільки на самого себе і на многочлени нульового степеня. Елемент x скінченного поля $GF(2^m)$ називається коренем многочлена $f(t)$, якщо $f(x) = 0$. Незвідний многочлен $f(t)$ називається примітивним, якщо його корені є примітивними елементами поля.

Примітивним п'ятичленом називається примітивний многочлен виду:

$$f(t) = t^m + t^l + t^j + t^k + 1, 0 < l < j < l < m.$$

Якщо x – корінь незвідного многочлена$f(t)$ степеня m, то елементи $(x^{m-1}, ..., 1)$ утворюють базис скінченного поля $GF(2^m)$ як векторного простору над полем $GF(2)$. Цей базис називається поліноміальним. Будь-який елемент основного поля однозначно виражається через елементи поліноміального базису. Найзручніше поліноміальний базис задавати примітивним многочленом [11], [12].

Висновки до розділу 1

У даному розділі було розглянуто основні відомості про приховані канали, дано їх класифікацію, наведено методи побудови прихованих каналів в криптосистемі з відкритим ключем DSA і алгоритмі ЦП на еліптичних кривих ГОСТ Р 34.10-2001.

2 ПРИХОВАНІ КАНАЛИ ПЕРЕДАЧІ ДАНИХ В АЛГОРИТМІ ЦП ДСТУ 4145-2002

ДСТУ 4145-2002 «Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка» – нині діючий національний стандарт України, що описує механізм формування та перевірки ЕЦП, що базується на властивостях групи точок еліптичної кривої над полем $GF(2^m)$. Введений в дію 28 грудня 2002 року, чинний від 1 липня 2003 року.

Розглянемо можливість побудови прихованих каналів в алгоритмі ЦП ДСТУ 4145-2002. Спочатку наведемо алгоритм формування ЦП в ДСТУ 4145-2002.

2.1 ДСТУ 4145-2002

Обчислення ключів цифрового підпису:

1) особистий ключ d обчислюється як випадкове ціле число. Якщо $d = 0$ генерується нове випадкове число, доки не буде виконуватися $d \neq 0$;

2) відкритий ключ ЦП – точка еліптичної кривої $Q = -dP$, де P – базова точка еліптичної кривої, d – особистий ключ.

Перевірка правильності цифрового підпису:

1) Перевірка правильності відкритого ключа Q.

– координати відкритого ключа ЦП належать основному полю – двійкові рядки довжини m;

– $Q \neq O$;

– (x_Q, y_Q) задовольняють рівняння еліптичної кривої алгоритму ЦП, тобто лежать на еліптичній кривій;

– $nQ = O$.

2) Перевірка правильності особистого ключа d:

обчислюється точка $Q^{'} = -dP$ еліптичної кривої, P – базова точка, d – особистий ключ. Особистий ключ є правильним тоді і тільки тоді, коли $Q^{'} = Q$.

Обчислення ЦП:

1) обчислюється геш-функція від повідомлення T – $H(T)$;

2) $H(T)$ перетворюється на елемент основного поля h і якщо $h = 0$, прийняти $h = 1$;

3) обчислюється цифровий передпідпис F_e за наступним алгоритмом:

а) обчислюється випадкове ціле число e;

б) обчислюється точка еліптичної кривої $R = (x_R, y_R) = eP$;

в) якщо координата $x_R = 0$ потрібно повернутися до кроку 1, інакше $F_e = x_R$.

4) обчислюється елемент основного поля $y = hF_e$;

5) елемент основоного поля y перетворюється на ціле число r і якщо $r = 0$, потрібно повернутися до кроку 3;

6) обчислюється ціле число $s = (e + dr) \bmod n$ і якщо $s = 0$, потрібно повернуться до кроку 3;

7) пара цілих чисел (r, s) перетворюється на цифровий підпис D довжини L_D.

Підписаним повідомленням є трійка (iH, T, D).

Перевірка ЦП.

1) За повідомленням T обчислюється функція гешування $H(T)$.

2) $H(T)$ ставиться у відповідність елемент основного поля h і якщо $h = 0$, приймається $h = 1$.

3) З цифрового підпису D знаходиться пара чисел (r, s).

4) Якщо не виконується хоча б одна з умов: $0 < r < n$, $0 < s < n$, алгоритм припиняється – підпис не вірний.

5) Обчислюється точка еліптичної кривої $R = (x_r, y_r) = sP + rQ$

6) Обчислюється елемент основного поля $y = hx_r$ і йому ставиться у

відповідність ціле число r'.

7) Якщо виконується $r = r'$ підпис вірний, інакше – не вірний.

2.2 Широкосмугові канали в ДСТУ 4145-2002

Розглянемо побудову широкосмугового каналу в даному алгоритмі ЦП. На кроці 4) досліджуваного алгоритму ЦП відбувається генерація випадкового числа e відповідно до наступного алгоритму.

Обчислення випадкового цілого числа

Довжина випадкового числа $L(e) < L(n)$, де n – порядок базової точки еліптичної кривої. Довжина випадкової послідовності – t.

1) Обчислити довжину $L(n)$ двійкового зображення числа n.

2) Обчислити мінімальне k, для якого $kt \geq L(n) - 1$.

3) За k звернень до генератора випадкових послідовностей формується двійковий рядок довжини kt. Перші $L(n) - 1$ елементів цієї послідовності формують випадковий двійковий рядок $R_{(L(n)-2)}, ..., R_0$ довжини $L(n) - 1$.

4) $e_i = R_i$ для $i = 0, ..., L(n) - 2$.

5) Знайти індекс j як найбільше i, для якого $e_i = 1$. Якщо такого індексу немає, прийняти $e = 0$ та припинити виконання алгоритму.

6) Випадковий рядок $R_{(L(n)-2)}, ..., R_0$ знищується.

Результатом виконання алгоритму є ціле число e зображене двійковим рядком $(e_j, ..., e_0)$.

Найпростішим випадком організації широкосмугового каналу є підміна випадкового числа e на приховане повідомлення. В такому випадку алгоритм формування ЦП не змінюється, а отримувач може обчислити приховане повідомлення знаючи ключ підпису d з наступної

рівності:

$$e = (s - dr) mod n.$$

Якщо ключ підпису невідомий, для знаходження e потрібно вирішити проблему дискретного логарифмування в групі точок еліптичної кривої.

В стандарті ДСТУ 4145-2002 наведено можливі варіанти для вибору початкових параметрів – степені розширення поля m, коефіцієнтів A і B еліптичної кривої E, порядку n базової точки. Для $m = 163$ $L(n) = 163$, отже верхня межа довжини прихованого повідомлення $L(M) = L(e) < 163$, що відповідає 20 байтам інформації. Для $m = 431$ $L(n) = 431$, отже $L(M) = L(e) < 431$ і за допомогою прихованого каналу можна передати до 53 байтів інформації.

Значення $m = 431$ є максимальним з можливих для ЕК в оптимальному і нормальному базисах. Тому максимальна довжина прихованого повідомлення – 53 байти. Якщо для перетворення повідомлення m на двійковий рядок використовувати 1-байтне кодування(наприклад, utf-8), по каналу можливо передати 53 символи. Але якщо зміст повідомлення передбачає тільки наявність кирилиці, для кодування кожного символу достатньо 5-ти біт. Тоді по каналу можна приховано передати до 86 символів.

Переваги широкосмугового каналу:

– можливість передачи великих повідомлень, тобто таких, довжина яких порівнянна в довжиною випадкового числа e;

– проста реалізація, що не потребує великої обчислювальної потужності чи додаткової пам'яті.

Недоліки даного каналу.

– Перевіряючому необхідно знати ключ підпису d. Тому даний канал можливо організувати тільки якщо підписувач повністю довіряє отримувачу і впевнений у відсутності можливості компрометації секретного ключа.

– При повторному передаванні прихованого повідомлення

відбувається компрометація секретного ключа d.

– На випадкове число e накладаються наступні обмеження:

1) на кроці 3) алгоритму формування ЦП координата x_R точки $R = (x_R, y_R) = eP$ не дорівнює 0, інакше потрібно переобчислити число e;

2) на кроці 5) алгоритму формування ЦП, якщо елемент основного поля y, який отримується з рівності $y = hF_e$, де $F_e = x_R$ при перетворенні на ціле число дорівнює 0, потрібно переобчислити число e;

3) на кроці 7) формування ЦП, якщо число $s = (e + dr) \bmod n$ дорівнює 0, потрібно переобчислити число e.

Розглянемо 2-й недолік більш детально. Якщо відправник двічі передасть по прихованому каналу однакове повідомлення, приховане в числі e, буде справедливою система:

$$\begin{cases} s_1 = e + dr_1(modn), \\ s_2 = e + dr_2(modn). \end{cases}$$

Звідки випливає рівність для знаходження числа d:

$$e = s_1 - dr_1 modn,$$
$$s_2 - s_1 - dr_1 + dr_2 modn,$$
$$dr_1 - dr_2 = s_1 - s_2 modn,$$
$$d = (s_1 - s_2)(r_1 - r_2)^{-1} modn.$$

Для запобігання данного недоліка можна використати метод підсолювання числа e – заміни частини його біт на випадкові. Одержувач, при обчисленні прихованого повідомлення, відкидає випадкові біти. В такому випадку суттєво зменшується пропускна здатність каналу, оскільки потрібно взяти велике число випадкових біт для уникнення атаки перебором. Припустимо, на підсолювання використовується 128 біт, тоді по каналу можливо передати до 37 байт інформації(при $m = 431$).

Для оптимізації данного підходу можна постійно змінювати позиції

випадкових біт за певним, відомим для одержувача, алгоритмом. Тоді кількість випадкових біт зменшиться, а пропускна здатність каналу зросте.

2.3 Вузькосмугові канали в ДСТУ 4145-2002

На практиці зручнішими для реалізації є вузькосмугові канали, оскільки вони не потребують розкриття секретного ключа підпису. Вони поділяються на ймовірнісні і детерміновані.

2.3.1 Ймовірнісний канал

Наведемо загальний алгоритм реалізації ймовірнісного каналу з [13].

1) Підписувач і одержувач домовляються про деяку секретну функцію $h_k()$, яка відображає підпис ζ або його частину (r або s) в деяку послідовність біт M, і про деякий спосіб кодування повідомлення такою послідовністю.

2) Підписувач генерує випадкові числа k, формує для кожного з них ЦП і обчислює значення функції $h_k()$ від отриманих підписів доки не отримає послідовність біт M, що відповідає прихованому повідомленню, або доки не вийде час, відведений на підпис.

3) Після перевірки підпису отримувач за допомогою секретного ключа K і алгоритму декодування послідовності M обчислює приховане повідомлення.

Розглянемо його побудову в ДСТУ 4145-2002.

Замість секретного ключа K використовується n різних великих

простих чисел. Функція $h_K()$ ставить у відповідність частині r підпису (r, s) послідовність M довжиною n біт, де i-й біт дорівнює «1», якщо r – квадратичний лишок за модулем i-го простого числа, і «0» – в іншому випадку. Вихідна послідовність M є прихованим повідомленням.

Будемо вважати, що за одиницю часу підписувач може генерувати L цифрових підписів. Перед побудовою прихованого каналу підписувач і отримувач мають обрати секретну функцію h_K і спосіб кодування прихованого повідомлення.

Тоді алгоритм побудови ймовірнісного каналу в ДСТУ 4145-2002 для передачі прихованого повідомлення довжини N виглядатиме так.

1. Підписувач генерує випадкове число e і обчислює ЦП (r,s).

2. Підписувач обчислює значениия функції h_k від частини r ЦП і ставить їй у відповідність «0» або «1» в залежності від обраного способу кодування.

3. Якщо отриманий біт відповідає відповідному біту повідомлення, він додається до вихідної послідовності M, випадкове число k передається отримувачу для декодування повідомлення. Інакше кроки 1-2 повторюються доки не буде отримано потрібний біт, але не більше L разів.

4. Якщо на попередньому кроці вдалося отримати потрібний біт, кроки 1-3 повторюються для наступного біту прихованого повідомлення. Інакше передача прихованого повідомлення припиняється – повідомлення не вдалося вбудувати у прихований канал.

Функція h_k має бути такою, що без знання секретного параметра k неможливо обрахувати прообраз функції.

Даний канал передачи зв'язку є 1-бітним, тобто за одну передачу інформації можна передати лише 1 біт прихованого повідомлення. Ймовірність того, що r буде квадратичним лишком за довільним простим числом n дорівняє $1/2$. Покажемо це:

Так як $x^2 \equiv (n-x)^2 \pmod{n}$, достатньо показати , що серед чисел $0^2, 1^2, ..., (\frac{n-1}{2})^2$ немає порівняних за модулем n.

(Від противного) Нехай існує таке $0 < y < \frac{n-1}{2}$, що $x^2 \equiv y^2(mod n)$, $y \neq x$ і $0 < x, y < \frac{n-1}{2}$. Так як $(x^2 - y^2)|n$, то $(x-y)(x+y)|n$. Оскільки n – просте і $0 < |x-y| < n$, маємо $x+y|n$, але $x+y \leq n-1$. Отже, припущення не вірне.

Оскільки P (r – квадратичний лишок(не лишок)) $= \frac{1}{2}$, то $P(n$ різних чисел - квадратичні лишоки(не лишки)) $= \frac{1}{2^n}$. Якщо підписувач за одиницю часу може генерувати тільки один підпис, тобто $L = 1$, ймовірність отримати повідомлення $P = \frac{1}{2^n}$ є дуже маленькою навіть для невеликих n. Тому в такому випадку буде не практично використовувати ймовірнісний канал. Але якщо $L > 1$, ймовірність отримати повідомлення $P = \frac{L}{2^n}$. Нехай $L = 2^l$, тоді $P = \frac{2^l}{2^n} = \frac{1}{2^{n-l}}$. В такому випадку при $L = 8$ з ймовірністю $\frac{1}{2}$ можливо передати повідомлення довжиною 4 біти. Зазначимо, що для передачі повідомлення з ймовірністю $P > \frac{1}{2}$ має виконуватися $l > n$.

Замість функції h_k можливо використовувати наступні функції.

– Функція гешування ГОСТ Р 34.11-2012 (замість вектора ініціалізації використовується число e).

– Піднесення числа r у секретний степінь e за модулем n. Цей спосіб базується на проблемі дискретного логарифмування в кільці лишків за простим модулем.

– Множення точки R, отриманої на кроці (3)(б) на секретне число e. Значенням секретної функції буде значення координати x отриманої точки.

В наведеному вище алгоритмі формування ймовірнісного каналу способом кодування вказано обчислення лишку (не лишку). Більш оптимальним з точки зору обчислювальної техніки є обрахунок по модулю 2. В такому випадку в якості результуючого біту, що додається до послідовності M, виступає останній біт результату секретної функції. Ймовірність отримання потрібного біту, як і раніше, дорівнює $1/2$.

2.3.2 Детермінований канал

Даний тип прихованих каналів дозволяє гарантовано передати приховане повідомлення, але потребує від отримувача великої обчислювальної потужності і додаткової пам'яті.

Нехай $H(x)$ – геш-функція з ДСТУ 4145-2002. У стандарті за замовчанням використовується функція гешування ГОСТ 34.311-95. Згідно чинного наказу Держспецзв'язку від 20 серпня 2012 року №1236/5/453 після 1 січня 2022 року стандарт повинен використовуватися з ДСТУ 7564-2014 (функція гешування «Купина») .

Алгоритм формування ПВП з використанням функції гешування і початково вектора ініціалізації d:

$$h_0 = H(d), h_i = H(h_(i-1)||d), i = 1, 2, ...$$

Наведемо алгоритм побудови детермінованого каналу в ДСТУ 4145-2002.

1. Підписувач генерує ключ для ПВП, побудованої за вище наведеним принципом і зберігає його у пам'яті, потім генерує n елементів ПВП, де n – число повідомлень, які можна передати по цому каналу.

2. На етапі передпідпису для кожного із n чисел обчислюється точка $R = eP$, і відправлюється отримувачу.

3. Підписувач починає формувати підпис за стандартним алгоритмом, але випадкові числа бере з ПВП, відновленої зі збережоного ключа. Замість точки R в алгоритмі використовується точка $R' = R + mP$, де m – приховане повідомлення, замість випадкового параметра e використовується число $e' = e + m$.

4. Після перевірки підпису отримувач додає до відповідної точки R точку P доки не отримає точку R'. За кількістю операцій додавання відновлюється повідомлення m.

Зауважимо, що $0 < m < 2^n$. Отже довжина прихованого повідомлення обмежується числом n повідомлень, які можна передати по прихованому каналу.

Множина точок R не дає ніякої інформації про «випадкові» числа, що використовуються для побудови підпису. Перевіряючий має точку $R = eP$, але щоб знайти e потрібно вирішити проблему дискретного логарифмування в групі точок еліптичної кривої і залежність між точкою і елементом ПВП занадто складна для знаходження випадкового числа. Таким чином отримувач не зможе обчислити секретний ключ d підписуючого. Суттєвим недоліком даного каналу є потреба передачі великої кількості інформації при передачі множини точок R і нерівномірність затраченого часу на декодування повідомлення в залежності від значення прихованого повідомлення.

Отримувач знає довжину прихованого повідомлення N, тому для нього приховане повідомлення m має таке обмеження: $2^{N-1} < m < 2^N - 1$ і на обчислення прихованого повідомлення потрібно від 0 до $2^{N-2} - 1$ операцій додавання точки ЕК.

Висновки до розділу 2

У даному розділі було розглянуто алгоритм формування ЦП в ДСТУ 4145-2002 і його особливості, можливість застосування прихованих каналів передачі даних в ДСТУ 4145-2002, наведено деякі оцінки можливої довжини прихованого повідомлення, ймовірность передачі прихованого повідомлення, алгоритми побудови різних типів прихованих каналів передачі даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРИХОВАНИХ КАНАЛІВ В АЛГОРИТМІ ЦП ДСТУ 4145-2002

У цьому розділі наведено приклади реалізації прихованих каналів в алгоритмі ЦП ДСТУ 4145-2002. Програмна реалізація виконана на мові програмування Java і міститься у додатку 1.

Пояснення до програмного коду:

– в пакеті field міститься клас Field, що відповідає полю $GF(2^m)$. Степінь розширення поля задається цілочисленим параметром m, примітивний поліном задається массивом розміру 5, що містить відповідні степені поліному. Наприклад, $f(t) = t^{163} + t^7 + t^6 + t^3 + 0$ відповідає массиву int[] polynom = 163,7,6,3,1,0. Також в пакеті field міститься клас FieldElement, об'єкт якого є елементом заданого поля Field. Цей клас містить реалізацію операцій додавання, множення, піднесення до степеня над полем $GF(2^m)$

– пакет eliptic curve містить класи ElipticCurve і Point. Клас ElipticCurve задає еліптичну криву за допомогою коефіцієнтів A і B, містить точку P – базову точку ЕК і її порядок N. У цьому класі реалізовані операції додавання і множення точок ЕК. Клас Point відповідає точці ЕК, містить поля x і y – відповідні координати точки ЕК.

– пакет hash містить класи Hash і Encryption, які реалізують геш-функцію як ітеративну функцію, побудовану на схемі Меркле-Дамгора. Значення хешу повертається 64-бітним числом типу long за допомогою функції Hash.getHash(byte[] M, long H0), де першим параметром є повідомлення, розбите на байти , а другим – вектор ініціалізації.

– клас SubliminalChannelInDSTU4145 містить реалізацію прихованих каналів. Зокрема функція broadbandChannel(BigInteger message, BigInteger subliminalMessage) містить реалізацію

широкосмугового каналу, на вхід приймає відповідно повідомлення для підпису і приховане повідомлення, функція determineSubliminalChannel(int n , BigInteger e, BigInteger message, List<BigInteger> subliminalMessages) реалізує детермінований прихований канал і приймає на вхід відповідно кількість прихованих повідомлень, які може передати канал, випадковий параметр для ГПВП, повідомлення для підпису, множину прихованих повідомлень.

3.1 Реалізація широкосмугового каналу

За допомогою широкосмугового прихованого каналу передамо повідомлення "Повідомлення". Для цього спочатку його потрібно закодувати кодуванням Windows-1251, або будь-яким іншим однобайтним кодуванням для підвищення пропускної спроможності каналу.

Отже, повідомлення "Повідомлення"перетворюється на набір символів $cfeee2b3e4eeecebe5ededff$, що відповідає числу 643521919513754116864615336 95 у шістнадцятковій системі числення.

Обчислення цифрового підпису.

Вважаємо, що всі перевірки згідно з розділом 12 ДСТУ підтверджують правильність відповідних данних. Нехай геш-функція від повідомлення, що підписується,

$$H = 09c9c44277910c9aaee486883a2eb95b7180166ddf73532eeb76edaef52247ff.$$

Тоді перетворимо результат обчислення функції гешування $H(T)$ на елемент основного поля. Для цього потрібно визначити кількість значуших біт числа як $min(m, L_H) = 163$, а інші(старші) біти відкинути. У результаті

цього перетворення отримаємо елемент поля

$$h = 03a2eb95b7180166ddf73532eeb76edaef52247f.$$

Далі обчислюється передпідпис F_e. Для цього потрібно згенерувати випадкове число e, але для передачі прихованого повідомлення замість числа e підставляється саме приховане повідомлення. Тобто $e = cfeee2b3e4eeecebe5ededff$.

Обчислюємо точку eP :

$$eP = (x_{eP}, y_{eP}) = (47f84191cdd9a46fb5855fa332a960b64bdae63e8,$$
$$34998609367757e8b2bb6830b36e4ea8bbf41a411).$$

Тоді F_e дорівнює координаті x_{eP} отриманої точки:

$$F_e = 47f84191cdd9a46fb5855fa332a960b64bdae63e8.$$

Обчислимо y як добуток елементів основного поля:

$$y = hF_e = 7f9535ca45ac31181d84c422381ed3cbe9af4b86d.$$

Згідно пункту 5.8 ДСТУ елемент y перетворюється на ціле число:

$$r = 7f9535ca45ac31181d84c422381ed3cbe9af4b86d.$$

Обчислюємо ціле число:

$$s = e + drmodn = 3c0654358be71e43ec904e341140ac8fea2b4addc.$$

Підписане повідомлення разом з підписом (r,s) відправляється перевіряючому. Алгоритм ЦП передбачає перетворення пари чисел (r,s) на цифровий підпис D згідно з пунктом 5.10 і його конкатенацію з початковим повідомленням T, але ми опустимо ці перетворення, адже поставлена задача цього не вимагає.

Отримувач має повідомлення T, цифровий підпис (r, s) і відкритий ключ Q.

Спочатку обчислюємо хеш-функцію від повідомлення. Нехай вона, як і в попередньому випадку, дорівнює:

$$H = 09c9c44277910c9aaee486883a2eb95b7180166ddf73532eeb76edaef52247ff.$$

Відповідний елемент основного поля:

$$h = 03a2eb95b7180166ddf73532eeb76edaef52247f.$$

Перевіряємо умову: $0 < r < n$, $0 < s < n$. Вона виконується, тому переходимо до обчислення точки еліптичної кривої:

$$R = sP + rQ = (x_R, y_R).$$

Спочатку обчислимо точку sP, її координати:

$$(x_sP, y_sP) = (2b4ce077c9ca3956e04df4c4f430ea24186ae4b0c,$$
$$6ab7455ddb6519d23a7d0e3e6b4bd96c9517bc88a).$$

Далі обчислимо точку rQ. Її координати:

$$(x_rQ, y_rQ) = (45008170641dc9a1a737c2127b6904149e117f537,$$
$$7b853570da2302f6478db2a300b51caa390855dc).$$

Результуюча точка R:

$$(x_R, y_R) = (47f84191cdd9a46fb5855fa332a960b64bdae63e8,$$
$$34998609367757e8b2bb6830b36e4ea8bbf41a411).$$

Обчислимо елемент основного поля $y' = hx_R$:

$$y' = 7f9535ca45ac31181d84c422381ed3cbe9af4b86d.$$

Підпис вірний, оскільки $y' = r$.

Тепер обчислимо приховане повідомлення за допомогою рівності $e = (s - dr) \bmod n$:

$$dr = 3c0654358be71e43df916008d2f1bdc12bcc6bfdd,$$

$$e = cfeee2b3e4eeecebe5ededff.$$

За допомогою даного прихованого каналу ми змогли передати і отримати приховане повідомлення $M =$ "Повідомлення".

3.2 Реалізація вузькосмугового каналу

3.2.1 Реалізація ймовірнісного каналу

Передамо по прихованому каналу 1 біт інформації, який може інтерпретуватися як «так» для «1» і «ні» для «0». Нехай потрібно передати «1» по ймовірнісному каналу.

Секретна ключова функція – визначення того, чи є частина r ЦП квадратичним лишком чи нелишком за модулем простого випадкового числа, а ключ для цієї функції – випадкове просте число e. Припустимо, що за одиницю часу підписуючий може генерувати не більше 3-х підписів.

Отже спочатку сгенеруємо просте число e за допомогою вбудованого в середовище генератора випадкових великих чисел:

$$e = 6e0a797a14145aa25cff935ffb862ae935467c74f.$$

Обчислюємо частину r ЦП за алгоритмом, встановленим ДСТУ:

$$Q = (x_Q, y_Q) = (57de7fde023ff929cb6ac785ce4b79cf64abdc2da,$$
$$3e85444324bcf06ad85abf6ad7b5f34770532b9aa),$$

$$eP = (x_eP, y_eP) = (979078dfa7b2e2e78aa69cce493ea2f8661ea8ad,$$
$$1bf4e79ec8f73cb0fd0b58afc290ee1cae049309e),$$

$$F = x_R = 979078dfa7b2e2e78aa69cce493ea2f8661ea8ad,$$

$$y = hF = 6272974b02d8a670826c2634a64df524a5f9627ea.$$

Перетворюєио елемент поля y на ціле число r:

$$r = 6272974b02d8a670826c2634a64df524a5f9627ea.$$

Тепер потрібно перевірити, чи є r квадратичним лишком. Перевіримо за критерієм Ейлера:

$$r^{(p-1)/2} = 6e0a797a14145aa25cff935ffb862ae935467c74e = -1.$$

Число e не підходить, тому потрібно його перерахувати. Якщо б за одиницю часу підписуючий міг генерувати тільки одне число, то вбудувати приховане повідомлення в такому випадку не вдалося.

Наступне випадкове число, отримане з генератора випадкових чисел:

$$e = 5383cfcbe0ff5f987140d230fab187143ce6829c1.$$

Обчислюємо частину r ЦП за алгоритмом, встановленим ДСТУ:

$$Q = (x_Q, y_Q) = (57de7fde023ff929cb6ac785ce4b79cf64abdc2da,$$
$$3e85444324bcf06ad85abf6ad7b5f34770532b9aa),$$

$$eP = (x_eP, y_eP) = (2ab6720698cf4f26608feb384274d72d30d9741b6,$$
$$24236d6ddd28035852511f835a2c47e084a74bb45),$$

$$F = x_R = 2ab6720698cf4f26608feb384274d72d30d9741b6,$$

$$y = hF = 9714aeee866e4c3840e8fa863a019ce3225e13bb.$$

перетворюєио елемент поля y на ціле число r:

$$r = 9714aeee866e4c3840e8fa863a019ce3225e13bb.$$

Тепер потрібно перевірити, чи є r квадратичним лишком. Перевіримо з акритерієм Ейлера:

$$r^{(p-1)/2} = 5383cfcbe0ff5f987140d230fab187143ce6829c0 = -1.$$

Оскільки знову отримали нелишок за модулем e, потрібно знову перераховувати e.

Наступне випадкове число, отримане з генератора випадкових чисел:

$$e = 5f9c46c46ddd9ac8f4bff98a664612610310a28f7.$$

Обчислюємо частину r ЦП за алгоритмом, встановленим ДСТУ:

$$Q = (x_Q, y_Q) = (57de7fde023ff929cb6ac785ce4b79cf64abdc2da,$$
$$3e85444324bcf06ad85abf6ad7b5f34770532b9aa),$$

$$eP = (x_eP, y_eP) = (1d4a7fafb77a5f32939df1060529e37bd7b95f719,$$
$$63ee0e3e91fc12115402a3c95043fbb20cb134dcd),$$

$$F = x_R = 1d4a7fafb77a5f32939df1060529e37bd7b95f719,$$

$$y = hF = 17969a2bb89e86fc7912572df94c13f3cfb18a75.$$

перетворюєио елемент поля y на ціле число r:

$$r = 17969a2bb89e86fc7912572df94c13f3cfb18a75.$$

Тепер потрібно перевірити, чи є r квадратичним лишком. Перевіримо з акритерієм Ейлера:

$$r^{(p-1)/2} = 1.$$

Оскільки r є квадратичним лишком, число $e = 5f9c46c46ddd9ac8f4bff98a664612610310a28f7$ підходить і його потрібно передати отримувачу для обчислення прихованого повідомлення.

Нехай підписувач може за встановлений на підпис час згенерувати 8 підписів. Тоді ймовірність отримання прихованого повідомлення довжини 4 біти дорівнює $8 * \frac{1}{2^4} = \frac{8}{16} = \frac{1}{2}$. Для ймовірності передачі повідомлення більше $\frac{1}{2}$, має виконуватись $n > \frac{l}{2} - 1$, n – кількість повідомлень у встановлений проміжок часу, l – довжина прихованого повідомлення у бітах.

3.2.2 Детермінований канал

Наведемо реалізацію детермінованого прихованого каналу.

Для оптимізації будемо використовувати 5-бітне кодування українського алфавіту, букві «а» відповідатиме рядок 00000, «я» – 11111, «г» не використовується. Спочатку відправник і одержувач домовляються про розмір прихованих повідомлень, нехай довжина у двійковому вигляді прихованого повідомлення дорівнює 21 (20 біт на повідомлення і перший біт – 1). Передамо по прихованому каналу слово «Криптографія». Для цього розділимо його на 4 частини «кри», «пто»,

«гра» і «фія» і закодуємо 5-бітним вище наведеним кодуванням. Шістнадцяткове представлення прихованих повідомлень: $M_1 = b669$, $M_2 = ceb1$, $M_3 = 8e60$, $M_4 = dd5f$.

Запускаемо ГПВП побудований за таким принципом:

$$h_0 = H(e), h_i = H(h_{i-1}||e).$$

Замість вектора ініціалізації обираємо випадкове число. Нехай:

$$e = 4a2f15b54ab97823ae4b21f1e27ad6f3f1ace632a.$$

Тоді результат геш-функції:

$$h_0 = 3db23f99123306fb,$$

$$h_1 = 31891fc5cf321db9.$$

Нехай секретний ключ ЦП і базова точка не змінилися, тоді координати точки Q:

$$Q = (x_Q, y_Q) = (57de7fde023ff929cb6ac785ce4b79cf64abdc2da,$$

$$3e85444324bcf06ad85abf6ad7b5f34770532b9aa).$$

Обчислимо точку $eP = h_1P$:

$$(x_eP, y_eP) = (73a4ae93802aae43b19163905d22acb98bb3c3390,$$

$$3c210d7fada3204a715c0265e122471e62eb0f537).$$

і відправимо її отримувачу. В алгоритмі будемо використовувати точку $eP' = eP + mP$:

$$(x_{eP'}, y_{eP'}) = (1434e7eddb8b12105b46e7bd1fe6d6c58c6f3ef3d,$$

$$7573f7320c1351967bd67629a8c1065d1f3841dc0).$$

Елемент основного поля F:

$$F_e = 1434e7eddb8b12105b46e7bd1fe6d6c58c6f3ef3d.$$

Обчислимо елемент основного поля: $y = hF_e$:

$$2ef46cb1bb81ca38596277213df1be111a8ecfd45.$$

Відповідно числа r і s дорівнюють:

$$r = 2ef46cb1bb81ca38596277213df1be111a8ecfd45,$$

$$s = 200021700c1924763d219f0e18899af5b5f7a6fcb.$$

На цьому алгоритм формування підпису завершується, підпис відправляється отримувачу.

Отримувач обчислює точки sP, rQ, $R = sP + rQ$:

$$(x_sP, y_sP) = (1e2a24679194e4735e92d0350434755f2ae848136,$$

$$4aee3cb3daf14f35769e6c8801aaf364e45aef30d),$$

$$(x_rQ, y_rQ) = (5a56d3b07012d758da5900d5bfab3a9c96694c4aa,$$

$$79410fb63fb06691381cbd5daeb3c2dfc1afafdf1),$$

$$(x_R, y_R) = (1434e7eddb8b12105b46e7bd1fe6d6c58c6f3ef3d,$$

$$7573f7320c1351967bd67629a8c1065d1f3841dc0).$$

Далі потрубно обчислити елемент основного поля $y^{'} = hx_R$:

$$y^{'} = 2ef46cb1bb81ca38596277213df1be111a8ecfd45.$$

Перетворимо елемент основного поля на відповідне число $r^{'}$ Отримали рівність: $r^{'} = r$, отже підпис правильний, переходимо до обчислення прихованого повідомлення.

Для обчислення прихованого повідомлення викликаємо функцію getSubliminalMessageFromEPinDetermineChannel з параметрами eP, eP'.

Результат функції – b669, затрачений час в наносекундах – 129286415752, відповідно в секундах – 129,3 секунди.

Наступне приховане повідомлення $M_2 = ceb1$. Отримані результати:

$$h_2 = 41e9b16ed9cd8200,$$

$$(x_eP, y_eP) = (64c61f3866a4e9073349a06d404ceba fe3b4b2579,$$

$$7eecdf10a7afd46c37117a9d86e602ede36b3f00),$$

$$(x_eP', y_eP') = (1e51b57cabddbe7f7f010955c801da47c42bde7a6,$$

$$4894cabf238b164ac1c9e0f38114a5fd13a46c81b).$$

Викликаємо функцію для обчислення прихованого повідомленя з відповідними параметрами eP, eP' і отримуємо число $ceb1$. Затрачений час: 145652973404 наносекунд = 145,65 секунд.

Далі передамо приховане повідомлення $M_3 = 8e60$. Отримані результати:

$$h_3 = eb93a98f364947aa,$$

$$(x_eP, y_eP) = (68f57e631d7a60c1acb6e3e53c7478202db348355,$$

$$7aa012d90450f79f826b2855ecd5c9b2ef343b3e),$$

$$(x_eP', y_eP') = (69c85d4457497f602feffaead4066a5136463c331,$$

$$1101f16e28d7c7ba4ff7a95110e6d4b6cf497034f).$$

Функція для обрахунку прихованого повідомлення повертає правильний результат 8e60, затрачений час – 100352087480 наносекунд = 100,35 секунд.

Останнє приховане повідомлення $M_4 = dd5f$. Отримані результати:

$$h_3 = 2eba1c7457e1c8bd,$$

$$(x_eP, y_eP) = (4be1cda0f3d2121ef7e1efa614a369e6bea69e276,$$

$$15d7fb402f9ca6614ffdc356fd85a2cd8e23ea7d5),$$

$$(x_eP', y_eP') = (59ed7ed441c20bf2f601ebe8ad7e6c511e545c11f,$$

$$55988a15a00e935001ab0647769a41c7c34f57056).$$

Час роботи функції для обчислення прихованого повідомлення – 154474378403 наносекунд =154,5 секунд.

3.2.3 Ліквідація прихованих каналів

Приховані канали будуються завдяки випадковому параметру e, тому для ліквідації прихованих каналів потрібно унеможливити підробку випадкового параметра. Для цього потрібна третя, контролююча, сторона. В такому разі випадковий параметр має генеруватися при участі підписуючого і контреллера, при чому останній не має отримати жодної інформації щодо результуючого випадкового числа e для уникнення компрометації секретного ключа d. В [8] Сіммонс запропонував алгоритм ліквідації прихованих каналів для DSA. Його адаптація для ДСТУ 4145-2002 виглядає так.

1) Підписувач генерує число e_1 і відправляє контроллеру значення точки e_1P.

2) Контролер обирає випадкове число e_2 і відправляє його підписувачу.

3) Підписувач обчислює число $e = e_1e_2 \pmod n$ і використовує його замість випадкового параметру в обчисленні ЦП.

4) Після перевірки підпису контроллер перевіряє рівність $hx_{e_2P} \equiv r \mod n$.

Дійсно якщо підписувач використав число e_2 при обчисленні ЦП, то

$$x_{e_2P} \equiv x_{e_1e_2P} \bmod n,$$

$$x_{e_2P} \equiv F \bmod n,$$

$$hx_{e_2P} \equiv hF \bmod n.$$

Інформація, відправлена контролеру не дає йому можливості визначити випадкове число, використане в алгоритмі формування ЦП . Якщо в одиницю часу підписувач може генерувати тільки один підпис, то цей алгоритм ліквідує приховані канали, але на практиці це майже ніколи не виконується. Інакше на кроці 2 контроллер зможе організувати ймовірнісний вузькосмуговий канал. Тому повністю ліквідувати приховані канали цим алгоритмом неможливо.

Висновки до розділу 3

У даному розділі було приведено приклади реалізації широкосмугового, вузькосмугово детермінованого і ймовірнісного каналів передачі даних в алгоритмі ЦП ДСТУ 4145-2002. Для ймовірнісного каналу було приведено оцінку ймовірності отримання повідомлення в залежності від відведеного на підпис часу або ж від встановленої кількості генерацій підпису в одиницю часу. Для детермінованого каналу було наведено залежність часу обчислення прихованого повідомлення від довжини прихованого повідомлення. В кінці розділу наведено алгоритм ліквідації прихованих каналів, який в деяких випадках може унеможливити передачу прихованих повідомлень, але повністю ліквідувати приховані канали не може.

ВИСНОВКИ

У ході виконання даної роботи був проведений аналіз опублікованих джерел за тематикою прихованих каналів в алгоритмах ЦП, який показав що приховані каналу існують у таких відомих алгоритмах ЦП як: RSA, DSA, ECDSA.

У першому розділі наведено алгоритми побудови прихованих каналів для алгоритму ЦП DSA і російського стандарту ЦП на еліптичних кривих ГОСТ Р 34.10-2001. На початку другого розділу наведено загальний алгоритм формування ЦП в сучасному стандарті ЦП України ДСТУ 4145-2002. Далі наведено алгоритми формування прихованих каналів різних типів в вище зазначеному стандарті, досліджено їх переваги і недоліки. Третій розділ містить практичну реалізацію прихованих каналів, наведено детальні приклади передачі прихованих повідомлень з усіма проміжними результатами. В кінці третього розділу наведено можливий алгоритм ліквідації прихованих каналів, який може ліквідувати прихований канал за деяких умов формування ЦП, але в загальному випадку повністю приховані канали не ліквідує.

Дана інформація може бути використана для розробленнях ефективних алгоритмів повної ліквідації прихованих каналів у разі порушення політики безпеки або для розроблення протоколів передачі сеансових ключів симметричних криптографічних систем на основі прихованих каналів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Butler W. Lampson «A note on the confinement problem» Communications of the ACM CACM Homepage table of contents archive, Volume 16 Issue 10,613-615pp, Oct. 1973

2. G.J. Simmons, «The prisoner's channel and the subliminal channel», in Advances in Cryptology, Crypto' 83, pp.51-67, Plenum Press, New York and London, 1984.

3. G.J. Simmons, «The history of subliminal channels», IEEE Jour. on sel. Areas Comm., Vol.16, No.4, pp.452-462, 1998.

4. X. Zhao and N. Li, «Reversible watermarking with subliminal channel,» in Proc. of the 10th International Workshop on Information Hiding (IH'08), Santa Barbara, California, USA, ser. Lecture Notes in Computer Science, vol. 5284. , pp. 118–131, May 2008.

5. G. J. Simmons, «Subliminal communication is easy using the DSA» in Proc. of the 10th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'93), Lofthus, Norway, ser. Lecture Notes in Computer Science, vol. 765, pp. 218–232, May 1993.

6. J.-M. Bohli, M. I. G. Vasco, and R. Steinwandt,«A subliminal-free variant of ECDSA,» in Proc. of the 8th International Workshop on Information Hiding (IH'06), Alexandria, Virginia, USA, ser. Lecture Notes in Computer Science, vol. 4437, pp. 375–387, July 2006.

7. G.J. Simmons, «A secure subliminal channel», in Advances in Cryptology, Crypto' 85, LNCS 218, pp.33-41, 1985.

8. G.J. Simmons, «Subliminal communication is easy using the DSA», in Proc. EUROCYPT 93, LNCS 765, pp.218-232, 1993.

9. «Subliminal Channels in Cryptographic Systems» Christian Backer 23.07.2009 Seminar Topic: Covert Channels and Embedded Forensics

10. «Скрытые каналы передачи информации в алгоритме цифровой

подписи ГОСТ Р 34.10-2001» М.И Атмашкин, С.В. Белим, Математические структуры и моделирование, 2011, вып. 22, ст. 101-113.

11. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка ДСТУ 4145 - 2002

12. А. В. Бессалов, А. Б. Телиженко «Криптосистемы на елиптических кривых» Киев, 2004

13. Kobara K., Imai H. On the Channel Capacity of Narrow-band Subliminal Channels In Proc. of ICICS 1999 Vol.1726 p.309-323

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Цей додаток містить програмну реалізацію прихованих каналів в ДСТУ 4145-2002 з реалізацією операцій додавання і множення в поліноміальному базисі точок ЕК.

А.1 Програмна реалізація операційдодавання і множення в поліноміальному базисі для точок ЕК над полем $GF(2^m)$

А.1.1 Пакет eliptic curve з класами Curve і Point

```
package eliptic_curve;

import field.FieldElement;
import lombok.Builder;

import java.math.BigInteger;

@Builder
public class ElipticCurve {
    public FieldElement A;
    public FieldElement B;
    public BigInteger N;
    public Point P;

    public Point add(Point p1, Point p2){
        if(p1.isNull()){
            return new Point(p2.getX(),p2.getY());
        }
```

```
        if(p2.isNull()){
            return new Point(p1.getX(),p1.getY());
        }


        FieldElement lyambda = (p1.getY().add(p2.getY()))

            .multiply((p1.getX().add(p2.getX())).inverse());

        FieldElement p3X = lyambda.pow2()

            .add(lyambda).add(p1.getX()).add(p2.getX())
            .add(A);
        FieldElement p3Y =  p1.getY().add(p3X)

            .add(lyambda.multiply(p3X.add(p1.getX())));
        return new Point(p3X,p3Y);
    }

    public Point doublePoint(Point p1){
        if(p1.isNull()){
            return Point.getZeroPoint();
        }
        FieldElement lyambda = p1.getX()
            .add(p1.getY().multiply(p1.getX().inverse()));

        FieldElement p2X = p1.getX().pow2()
            .add(B.multiply(p1.getX().pow2().inverse()));

        FieldElement p2Y = p1.getX().pow2().add(p2X
            .multiply(lyambda.add(FieldElement.getOne())));
```

```
        return new Point(p2X,p2Y);
    }

    public Point multiply(Point p1, BigInteger e){

        char[] eInCharArray = e.toString(2).toCharArray();
        int[] eInBitArray = new int[eInCharArray.length];

        for(int i = 0 ; i < eInCharArray.length; i++){
            eInBitArray[i] = Integer.valueOf
                    (eInCharArray[eInCharArray
                    .length-1-i]+"");
        }

        Point r = new Point();
        Point p2 = new Point(p1.getX(),p1.getY());

        for (int i = 0 ; i < eInBitArray.length ; i++){
            if(eInBitArray[i] == 1){
                r = add(r,p2);
            }
            p2 = doublePoint(p2);
        }
        return r;
    }

    public Point getInverse(Point p1){
        return new Point(p1.getX(),p1.getX().add(p1.getY()));
    }
}
```

```
package eliptic_curve;

import field.FieldElement;

import java.math.BigInteger;

public class Point {
    private FieldElement x;
    private FieldElement y;

    public Point() {
        x = new FieldElement();
        y = new FieldElement();
    }

    public Point(FieldElement x, FieldElement y) {
        this.x = x;
        this.y = y;
    }

    public Point(BigInteger x, BigInteger y){
        this.x = new FieldElement(x);
        this.y = new FieldElement(y);
    }
    public static Point getZeroPoint(){
        return new Point();
    }

    public FieldElement getX() {
        return x;
```

```
    }

    public void setX(FieldElement x) {
        this.x = x;
    }

    public FieldElement getY() {
        return y;
    }

    public void setY(FieldElement y) {
        this.y = y;
    }

    @Override
    public String toString() {
        return "eliptic_curve.Point{" +
                "x=" + x +
                ", y=" + y +
                '}';
    }

    public boolean isNull() {
        Point point = new Point();
        return point.getX().cmp(x)==0&&

            point.getY().cmp(y)==0;
    }
}
```

A.1.2 Пакет field з класами Field і FieldElement

```
    package field;

public class Field {
    public  int m = 163;
    public  int[] polynom = {163,7,6,3,0};

    public Field() {
    }

    public Field(int m, int[] polynom) {
        this.m = m;
        this.polynom = polynom;
    }
}

package field;

import java.math.BigInteger;
import java.util.Arrays;

public class FieldElement {
    public static final Field field = new Field();
    public int[] element;
    public int length;

    public FieldElement(int[] element) {
        this.element = element;
```

```
        this.length = getLength();
    }

    public FieldElement() {
        element = new int[field.m];
        length = 0;
    }

    public FieldElement(int[] element, int length) {
        this.element = element;
        this.length = length;
    }

    public FieldElement(BigInteger b1){
        int length = b1.bitLength();

        if(length > field.m){
            throw  new IllegalArgumentException("length
                > "+field.m);
        }

        char[] b1CharArray = b1.toString(2).toCharArray();

        int[] b1BitArray = new int[field.m];

        for(int i = 0 ; i < length;i++){
            b1BitArray[i] = Integer.valueOf(

                b1CharArray[length-1-i]+"");
        }
```

```
        this.length = length;
        this.element = b1BitArray;
    }

    public static FieldElement getFieldElement(BigInteger b1){
        int length = b1.bitLength();

        if(length<=field.m){
            return new FieldElement(b1);
        }
        char[] b1InFieldCharArray = Arrays.copyOfRange
            (b1.toString(2).toCharArray()
            ,length-field.m-1,length-1);

        int[] b1BitArray = new  int[field.m];

        for(int i = 0 ; i < field.m;i++){
            b1BitArray[i] = Integer.valueOf
                (b1InFieldCharArray[field.m-1-i]+"");
        }

        return new FieldElement(b1BitArray);
    }

    public BigInteger getBigInteger(){
        char[] b1CharArray = new char[field.m];
        for(int i = 0 ; i < field.m ; i++){
            b1CharArray[i] =
(char)(element[field.m-i-1]+'0');
        }
        return new BigInteger(String.valueOf(b1CharArray),2);
```

```
}
public int getLength(){
    int length;
    for(length = field.m-1; length>-1; length--){
        if(element[length] == 1)
            break;
    }
    return length+1;
}

public int getLastPosition(){
    return getLength()-1;
}

public static FieldElement getOne(){
    int[] e3 = new int[field.m];
    e3[0] = 1;
    return new FieldElement(e3);
}

public static FieldElement getTwo(){
    int[] e3 = new int[field.m];
    e3[1] = 1;
    return new FieldElement(e3);
}

public static FieldElement getZero(){
    return new FieldElement();
}

@Override
```

```
public String toString() {
    return "field.FieldElement{" +
            "element=" + Arrays.toString(element) +
            '}';
}



public FieldElement add(FieldElement e2){
    FieldElement e3 = new FieldElement();
    for(int i = 0 ; i < field.m; i++){
        e3.element[i] = element[i]^e2.element[i];
    }
    return e3;
}

public FieldElement multiply(FieldElement e2){
    int[] e3 = new int[2*field.m];
    FieldElement e4 = new FieldElement();
    for(int i = 0 ; i < field.m; i++){
        if(e2.element[i] == 1){
            for(int j = 0 ; j<field.m;j++){
                e3[i+j] = e3[i+j]^element[j];
            }
        }
    }
    for(int i = 2*field.m-1; i>field.m-1; i--){
        if(e3[i] ==1){
            for(int j = 0 ; j < 5 ; j++){
                e3[i - (field.polynom[0]-field
                .polynom[j])] = e3[i - (field
```

```
                .polynom[0] - field.polynom[j])]^0x1;
            }
        }
    }
    for(int i = 0; i < field.m; i++){
        e4.element[i] = e3[i];
    }
    return e4;
}

public FieldElement pow2(){
    int[] e2 = new int[2*field.m];
    e2[0] = element[0];

    for(int i = 1 ; i < field.m;i++){
        e2[i*2] = element[i];
    }
    for(int i = 2*field.m-1; i>field.m-1; i--){
        if(e2[i] ==1){
            for(int j = 0 ; j < 5 ; j++){
                e2[i - (field.polynom[0]
                -field.polynom[j])] = e2[i - (field
                .polynom[0] - field.polynom[j])]^0x1;
            }
        }
    }
    FieldElement e3 = new FieldElement();
    for (int i = 0 ; i < field.m ; i++){
        e3.element[i] = e2[i];
    }
    return e3;
```

```
}

public FieldElement pow(FieldElement e2){
    FieldElement e3 = this;
    FieldElement result = FieldElement.getOne();
    for(int i = 0 ; i < e2.getLength() ; i++){
        if(e2.element[i]==1){
            result = result.multiply(e3);
        }
        e3 = e3.pow2();
    }
    return result;
}

public FieldElement inverse(){
    BigInteger exponent = BigInteger.valueOf(2)
        .pow(field.m).subtract(BigInteger.valueOf(2));
    FieldElement exp = new FieldElement(exponent);
    return this.pow(exp);
}

public FieldElement shiftRight(int shift){
    FieldElement e2 = new FieldElement();
    for(int i = shift; i < field.m; i++){
        e2.element[i] = element[i-shift];
    }
    return e2;
}

public FieldElement sub(FieldElement e2){
    int g = this.cmp(e2);
```

```
        FieldElement e3 = new FieldElement();
        if(g!=-1){
            int n = 0 , borrow = 0;
            for(int i =  0; i < field.m; i++){
                n =element[i] - e2.element[i] - borrow;
                if(n >=0){
                    e3.element[i] = n;
                    borrow = 0;
                }else {
                    e3.element[i] = n&0x1;
                    borrow = 1;
                }
            }
        }
        return e3;
    }

    public  int cmp(FieldElement e2){
        int i = field.m-1;
        while (element[i] == e2.element[i]) {
            i--;
            if(i == -1)
                return 0;
        }
        if(element[i] > e2.element[i])
            return 1;
        return -1;

    }
}
```

A.2 Программна реалізація геш-функції

```
    package hash;

import java.io.BufferedReader;
import java.nio.ByteBuffer;
import java.nio.channels.ByteChannel;

public class Encryption {
    private long M;
    private int L;
    private int R ;
    private long K_Key ;
    private int[] K_KEYS = new int[4];

    private Encryption(long M, long K){
        this.M = M;
        this.K_Key = K;
        initK_KEYS();
        initTABLE_S();
        SplitMessageIntoLR();
    }
    public final static String[] TABLE_S_String = {"99", "7c", "77"
    , "7b","f2", "6b", "6f", "c5", "30", "01", "67", "2b", "fe"
    , "d7", "ab", "76","ca", "82", "c9", "7d", "fa", "59", "47",
    "f0", "ad", "d4", "a2", "af", "9c", "a4", "72", "c0","b7",
    "fd", "93", "26", "36", "3f", "f7", "cc", "34", "a5", "e5",
    "f1", "71", "d8", "31", "15","04", "c7", "23", "c3","18",
    "96", "05", "9a", "07", "12", "80", "e2", "eb", "27", "b2",
```

```
"75","09", "83", "2c", "1a", "1b", "6e","5a", "a0", "52",
"3b", "d6", "b3", "29", "e3", "2f", "84","53", "d1", "00",
"ed", "20", "fc", "b1", "5b", "6a", "cb", "be", "39", "4a",
"4c", "58", "cf", "d0", "ef", "aa", "fb", "43", "4d", "33",
"85", "45", "f9", "02", "7f", "50", "3c", "9f", "a8","51",
"a3", "40", "8f", "92", "9d", "38", "f5", "bc", "b6", "da",
"21", "10", "ff", "f3", "d2","cd", "0c", "13", "ec", "5f",
"97", "44", "17","c4", "a7", "7e", "3d", "64", "5d", "19",
"73","60", "81", "4f", "dc","22", "2a", "90", "88", "46",
"ee", "b8", "14", "de", "5e", "0b", "db","e0", "32", "3a",
"0a", "49", "06", "24", "5c", "c2", "d3", "ac", "62", "91",
"95", "e4", "79", "e7", "c8", "37", "6d", "8d", "d5", "4e",
"a9", "6c", "56", "f4", "ea", "65", "7a", "ae", "08", "ba",
"78", "25", "2e", "1c", "a6", "b4", "c6", "e8", "dd", "74",
"1f", "4b", "bd", "8b", "8a", "70", "3e", "b5", "66", "48",
"03", "f6", "0e", "61", "35", "57", "b9", "86", "c1", "1d",
"9e","e1", "f8", "98", "11", "69", "d9", "8e", "94", "9b",
"1e", "87", "e9", "ce", "55", "28", "df","8c", "a1", "89",
"0d", "bf", "e6", "42", "68", "41", "99", "2d", "0f", "b0",
"54", "bb", "16"};

public static int[] TABLE_S = new int[256];

public static void initTABLE_S(){
    for(int i = 0 ; i < 256; i++){
        TABLE_S[i] = Integer.valueOf(TABLE_S_String[i],16);
    }
}
public void initK_KEYS() {
    int k = (int)(K_Key>>32);
    int k2 = (int)K_Key;
```

```
    K_KEYS[0] = ((k&0xff)<<24)  + ((k&0xff00) << 8)
        +((k&0xff0000) >> 8) +((k>>24)&0xff);
    K_KEYS[1] = ((k2&0xff)<<24)  + ((k2&0xff00) << 8)
        +((k2&0xff0000) >> 8) +((k2>>24)&0xff);
    K_KEYS[3] = ~K_KEYS[0];
    K_KEYS[2] = ~K_KEYS[1];

}


public void SplitMessageIntoLR()
{
    int l = (int)(M>>32);
    int r = (int)M;
    L = ((l&0xff)<<24)  + ((l&0xff00) << 8)
        +((l&0xff0000) >> 8) +(l>>24);
    R = ((r&0xff)<<24)  + ((r&0xff00) << 8)
        +((r&0xff0000) >> 8) +(r>>24);
}
public static long GetCrypto(long M, long K){

    Encryption e = new Encryption(M,K);
    e.Rounder();
    long res = Long.rotateLeft((e.R&0xff),56)
        + Long.rotateLeft(e.R&0xff00,40) +Long.
            rotateLeft(e.R&0xff0000,24) +Long.rotateLeft
            ((e.R>>24)&0xff,32)+
        Long.rotateLeft((e.L&0xff),24)  + ((e.L&0xff00) << 8)
        +((e.L&0xff0000) >> 8) +((e.L>>24)&0xff);
    return   res;
}
```

```
    public void Rounder(){
        for(int i = 1 ; i < 5 ; i++){
            Raund(i);
        }
    }
    public void Raund(int i){
        int R_i = R;
        R = L;
        L = F(K_KEYS[i-1],R_i)^R;
    }

    public int F(int K, int R){
        return Integer.rotateLeft(S(K^R),13);
    }

    public int S(int K){
        byte[] bytes = ByteBuffer.allocate(4)
            .putInt(K).array();
        for(int i = 0 ; i < bytes.length; i++) {
            bytes[i] = (byte) TABLE_S[Byte
                .toUnsignedInt(bytes[i])];
        }
        return ByteBuffer.wrap(bytes).getInt();
    }
}

package hash;

import java.math.BigInteger;
import java.nio.ByteBuffer;
import java.util.Arrays;
```

```
public class Hash {

    public final static int size = 214748363;
    public  final  static BigInteger FF =
        new BigInteger("ffffffffffffffff",16);

    public static byte[] padding(byte[]  b){
        int padding = 8-b.length-1;
        byte[] res = new byte[8];
        for( int i = 0 ; i< b.length;i++){
            res[i] = b[i];
        }
        if(b.length<8) {
            res[b.length] = (byte) 128;
            while (padding > 0) {
                res[8 - padding] = 0;
                padding--;
            }

        }
        return res;
    }

    public static long getHash(byte[] M, long H0){
        long H = H0;
        byte[] m ;
        long h ;

            for (int i = 0; i < M.length; i += 8) {
```

```
                if (i + 8 < M.length) {
                    m = Arrays.copyOfRange(M, i, i + 8);
                } else {
                    m = padding(Arrays.copyOfRange(M, i,
                            M.length));
                }
                h = H;
                H = iterationFunctionG(ByteBuffer
                    .wrap(m).getLong(), h);
            }
        return  H;
    }

    private static long iterationFunctionG(long m, long h) {
        return Encryption.GetCrypto(m,h)^m;
    }
}
```

A.3 Програмна реалізація призованих каналів

A.3.1 Клас SubliminalChannelInDSTU4145

```
    import eliptic_curve.ElipticCurve;
import eliptic_curve.Point;
import field.FieldElement;
import hash.Hash;
import org.apache.commons.lang3.ArrayUtils;
```

```
import java.math.BigInteger;
import java.nio.ByteBuffer;
import java.util.List;
import java.util.Random;

public class SubliminalChannelInDSTU4145 {
    public static final BigInteger TWO   =
        BigInteger.valueOf(2);
    public  ElipticCurve curve;

    public SubliminalChannelInDSTU4145(
        ElipticCurve curve){
        this.curve = curve;
    }

    public void determineSubliminalChannel(int n ,
        BigInteger e, BigInteger message, List<BigInteger>
            subliminalMessages,int subLength){

        long h0 = Hash.getHash(e.toByteArray(),0l);
        print("init h0",Long.toHexString(h0));
        byte[]  arrayForHash = null;

        Point Q = null;
        Point ePinit = null;
        Point eP = null;
        Point sP = null;
        Point rQ = null;
        Point R = null;

        BigInteger d = null;
```

```
BigInteger r = null;
BigInteger s = null;
BigInteger e1 = null;
BigInteger subMes = null;

FieldElement h = null;
FieldElement F = null;
FieldElement y = null;
FieldElement y1 = null;


for (BigInteger subliminalMessage:
        subliminalMessages) {
    arrayForHash = ArrayUtils.addAll(ByteBuffer.allocate(8)
        .putLong(h0).array(),e.toByteArray());
    h0 = Hash.getHash(arrayForHash,0l);
    print("h0", Long.toHexString(h0));
    ePinit = curve.multiply(curve.P,new BigInteger
        (Long.toHexString(h0),16));

    d = getD();
    h = getH(message);

    Q = curve.getInverse(curve.multiply(curve.P,d));
    print("Q",Q);

    eP = curve.add(ePinit,curve.multiply(curve.P,
        subliminalMessage));
    print("eP",eP);

    F = eP.getX();
```

```
print("F",F);
y = h.multiply(F);
print(" y ",y);

r = y.getBigInteger();
s = (new BigInteger(Long.toHexString(h0),16)
    .add(subliminalMessage)).add(d.multiply(r))
    .mod(curve.N);

print("s ",s.toString(16));
print("r ",r.toString(16));

sP = curve.multiply(curve.P,s);
print("sP" ,sP);

rQ = curve.multiply(Q,r);
print("rQ",rQ);

R = curve.add(curve.multiply(curve.P,s)
    ,curve.multiply(Q,r));
print("R",R);

y1 = h.multiply(R.getX());
print("y_tilda",y1);

print("check sign:",y1.getBigInteger()
    .equals(r));

print("get subliminal message");
subMes = getSubliminalMessageFromEPinDetermineChannel
    (ePinit,eP,subLength);
```

```
            print("subliminal message after algorithm",subMes);

            n--;
        }
    }

    public BigInteger getSubliminalMessageFromEPinDetermineChannel
        (Point ePinit, Point eP, int subLength){
        BigInteger message = BigInteger.valueOf(2^(subLength-1));
        print("in get fuction ePinit",ePinit);
        print("in get function eP ",eP);

        print("time nano",System.nanoTime());
        print("time",System.currentTimeMillis());

        while (true){
            ePinit = curve.add(curve.P,ePinit);
            message = message.add(BigInteger.ONE);
            if(ePinit.getX().getBigInteger()
            .equals(eP.getX().getBigInteger())&&ePinit.getY()
            .getBigInteger().equals(eP.getY().getBigInteger()))
                break;
        }
        print("time nano",System.nanoTime());
        print("time",System.currentTimeMillis());
        print("calculated message",message.toString(10));
        return message;
    }

    public Point startPointForGettingDetermineMassage
        (Point P,int n){
```

```
    return curve.multiply(P,new BigInteger((1<<n)+""));
}


public void getNumbersForDetermineSubliminalChannel
    (int n ,BigInteger e){
    long h0 = Hash.getHash(e.toByteArray(),0l);
    byte[]  arrayForHash = null;
    Point eP = null;
    Long h0Long = null;
    while(n>0){
        arrayForHash = ArrayUtils.addAll(ByteBuffer.allocate(8)
        .putLong(h0).array(),e.toByteArray());
        h0 = Hash.getHash(arrayForHash,0l);
        print("h0", Long.toHexString(h0));
        eP = curve.multiply(curve.P,new BigInteger(Long
            .toHexString(h0),16));
        print("eP",eP);
        n--;
    }
}

public void narrowbandChannel(int message){{
    String messageString = BigInteger.valueOf(message)
        .toString(2);
    char[] messageInArray = messageString.toCharArray();
    for(char c :
    messageInArray){
        print("Bit in subliminal message ",c);
        if(c =='0')
            getBitInFirstNarrowbandChannel(null,0);
```

```
        else
            getBitInFirstNarrowbandChannel(null,1);
    }
}

}
public BigInteger getBitInFirstNarrowbandChannel
    (BigInteger message, int subliminalbit){
    BigInteger e = null;
    BigInteger r = null;
    int counter = 0;
    do{
        do{
            e = BigInteger.probablePrime(163, new Random());
        }while (!e.isProbablePrime(50));
        print("e maybe right",e.toString(16));
        r = GetSignOnlyWithR(e,message);
        counter++;
    }while (!checkIfRightBit(isQuadricResudo(r,e),subliminalbit));

    BigInteger s = e.add(getD().multiply(r)).mod(curve.N);
    print("s ",s.toString(16));
    print("e in subliminal message",e.toString(16));
    print("counter",counter);
    return e;
}

public boolean checkIfRightBit(boolean isQuadric,
                               int suliminalBit){
    if(isQuadric&&suliminalBit==1) return true;
    if(!isQuadric&&suliminalBit==0) return true;
```

```
    return false;
}

public BigInteger GetSignOnlyWithR(BigInteger e,
                            BigInteger message){
    BigInteger d = getD();
    FieldElement h = getH(message);

    Point Q = curve.getInverse(curve.multiply(curve.P,d));
    print("Q",Q);
    Point eP = curve.multiply(curve.P,e);
    print("eP",eP);

    FieldElement F = eP.getX();
    print("F",F);
    FieldElement y = h.multiply(F);
    print(" y ",y);
    return y.getBigInteger();
}
public boolean isQuadricResudo(BigInteger b,
                            BigInteger module){
    BigInteger bEulerTest = b.modPow(module.
        subtract(BigInteger.ONE).divide(TWO),module);
    print("euler test",bEulerTest);
    return bEulerTest.equals(BigInteger.ONE);
}

public BigInteger broadbandChannel(BigInteger message,
                        BigInteger sumliminalMessage){
    BigInteger d = getD();
    FieldElement h = getH(message);
```

```
Point Q = curve.getInverse(curve.multiply(curve.P,d));
print("Q",Q);
BigInteger e = sumliminalMessage;

Point eP = curve.multiply(curve.P,e);
print("eP",eP);

FieldElement F = eP.getX();
FieldElement y = h.multiply(F);
print(" y ",y);

BigInteger r = y.getBigInteger();
BigInteger s = e.add(d.multiply(r)).mod(curve.N);

print("s ",s.toString(16));
print("r ",r.toString(16));

Point sP = curve.multiply(curve.P,s);
print("sP" ,sP);

Point rQ = curve.multiply(Q,r);
print("rQ",rQ);

Point R = curve.add(curve.multiply(curve.P,s),curve.
    multiply(Q,r));
print("R",R);

FieldElement y1 = h.multiply(R.getX());
print("y_tilda",y1);
```

```
    print("check sign:",y1.getBigInteger().equals(r));

    BigInteger e1 = s.subtract(d.multiply(r)).mod(curve.N);
    print("dr ",d.multiply((r)).mod(curve.N).toString(16));
    print("subliminal message",e1.toString(16));

    return e1;
}

private BigInteger getD(){
    return new BigInteger
        ("183F60FDF7951FF47D67193F8D073790C1C9B5A3E",16);
}

private FieldElement getH(BigInteger message){
    return new FieldElement(new BigInteger
        ("3A2EB95B7180166DDF73532EEB76EDAEF52247FF",16));
}

public void print(String message,Object o){
    System.out.println(message+" "+o);
}

public void print(String message, Point p){
    System.out.println(message+" "+p.getX().
        getBigInteger().toString(16)+" "+p.getY().
            getBigInteger().toString(16));
}

public void print(String message, FieldElement f){
    System.out.println(message+" "+f.
```

```
            getBigInteger().toString(16));
    }

    public void print(String message){
        System.out.println(message);
    }

    public void print(String message, BigInteger b){
        System.out.println(message+" "+b.toString(16));
    }

}
```

A.3.2 Клас Controller

```
    import eliptic_curve.ElipticCurve;
import eliptic_curve.Point;
import field.FieldElement;

import java.math.BigInteger;
import java.util.ArrayList;
import java.util.List;

public class Controller {
    public static void main(String[] args) {
        ElipticCurve curve = ElipticCurve.builder()
                .A(FieldElement.getOne())
```

```
                .B(new FieldElement(new BigInteger
                ("5FF6108462A2DC8210AB403925E638A19C1455D21"
                , 16)))
                .N(new BigInteger
                ("400000000000000000002BEC12BE2262D39BCF14D"
                ,16))
                .P(new Point(new FieldElement(new BigInteger
                ("72D867F93A93AC27DF9FF01AFFE74885C8C540420"
                , 16)),new FieldElement(new BigInteger
                ("0224A9C3947852B97C5599D5F4AB81122ADC3FD9B"
                , 16)))).build();
        SubliminalChannelInDSTU4145 dstu =
            new SubliminalChannelInDSTU4145(curve);
        dstu.broadbandChannel(null,new BigInteger("af34af",16));
        dstu.narrowbandChannel(31);

        List<BigInteger> subMessages = new ArrayList<BigInteger>();
        subMessages.add(new BigInteger("1011011001101001",2));
        subMessages.add(new BigInteger("1100111010110001",2));
        subMessages.add(new BigInteger("1000111001100000",2));
        subMessages.add(new BigInteger("1101110101011111",2));

        subMessages.stream().forEach(e -> System.out.println(
            e.toString(16)));

        dstu.determineSubliminalChannel(4,new BigInteger
        ("4a2f15b54ab97823ae4b21f1e27ad6f3f1ace632a",16)
            ,null,subMessages,16);
    }
}
```